View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by NLR Reports Repository

Nationaal Lucht- en Ruimtevaartlaboratorium

National Aerospace Laboratory NLR

NLR-TP-2003-475

Optimisation of Airport Taxi Planning

J.W. Smeltink[1], M.J. Soomer[1,2], P.R. de Waal[1] and R.D. van der Mei[2,3]

1 National Aerospace Laboratory NLR, Amsterdam, the Netherlands

2 Vrije Universiteit, Faculty of Exact Sciences, Amsterdam, the Netherlands

3 TNO Telecom, Center of Excellence Quality of Service, Leidschendam, the Netherlands

This report is based on an article to be published in the European Journal of Operations Research.

This report may be cited on condition that full credit is given to NLR and the authors.

Customer:	National Aerospace Laboratory NLR
Working Plan number:	I.1.A.2
Owner:	National Aerospace Laboratory NLR
Division:	Information and Communication Technology
Distribution:	Unlimited
Classification title:	Unclassified
	September 2003

Approved by author:	Approved by project manager:	Approved by project managing department:
2/10	2/10	2/10

Contents

2 Tables
5 Figures

(24 pages in total)

List of used symbols and abbreviations

A	set of arcs of a graph
A^{hold}	set of arcs used for holding
$\mathcal{A}$	set of aircraft
$\mathcal{A}^0$	set of aircraft extended with dummy aircraft
$\mathcal{A}^{\text{arr}}$	set of arriving aircraft
$\mathcal{A}^{\text{dep}}$	set of departing aircraft
ATA	Actual Time of Arrival
ETA	Estimated Time of Arrival
d_{sep}	minimum separation distance between aircraft
$d_{\text{iuv}}^{\min}$	minimum time for aircraft i to taxi arc (u, v)
$d_{\text{iuv}}^{\max}$	maximum time for aircraft i to taxi arc (u, v)
G	a graph
$l(u, v)$	length of arc (u, v)
m	size of sliding window
MIP	Mixed Integer Programming
OBT	(estimated) Off-Block Time
R_i	route of aircraft i
T	length of time interval
t_{iu}	time aircraft i reaches node u
TTD	Target Time of Departure
V	set of nodes of a graph
y_{iju}, z_{iju}	variables to determine order of aircraft i and j at node u

Summary

Over the past few decades the amount of air traffic has been growing dramatically. As many improvements have been achieved in enlarging the en-route capacity, airports are becoming the air traffic bottleneck. This raises the need for airports to optimise the efficiency of their logistic processes, such as arrival and departure management, stand allocation, and taxi planning. In this paper we focus on taxi planning. We present a model and algorithms to improve efficiency of the taxi process. Numerical results with real data of Amsterdam Airport Schiphol demonstrate that the algorithms lead to significant improvements of the efficiency within reasonable time.

This page is intentionally left blank.

1 Introduction

Over the past few decades air traffic has experienced a tremendous growth. To accommodate this growth while maintaining the safety requirements, improvements have to be made. Recently, a lot of improvements have been achieved in enlarging the en-route traffic capacity. As a result, the air traffic bottleneck tends to shift from the en-route capacity nowadays to the capacity of the airports. This development has forced airports, airlines, and Air Traffic Control service providers to improve the efficiency of the individual airport processes such as arrival [9], departure [6], stand allocation [1] and turn-around management, but also to devote effort to the integration of all these processes and systems: Collaborative Decision Making [3]. Currently, the control of ground movements at airports (taxiing aircraft) depends primarily on visual guidance, which is highly complicated under low visibility conditions. In reduced visibility conditions, the handling of taxiing aircraft is the bottleneck constraining the airport capacity. Furthermore, taxiing aircraft raise the fuel emission levels. These emissions have become a major environmental problem. By reducing unnecessary waiting times, these emission levels can be reduced. These observations have raised the need to model and optimise the traffic flows of taxiing aircraft, maintaining, of course, the current level of safety. Literature on the planning of the taxiing aircraft focuses mainly on simulation of the taxi flows [2, 11] or estimation of taxi times [3, 7]. This paper presents a model and algorithms for optimal taxi planning: the planning and scheduling of individual aircraft to provide safe, expeditious, and efficient movement from its current position (gate or runway) to its intended location (gate or runway). The algorithms have been applied to optimise taxi planning for Amsterdam Airport Schiphol. The results demonstrate that the algorithms lead to significant improvements of the taxi planning within reasonable time, and as such are practically useful for tactical planning of the aircraft.

2 Taxi process

The taxi process for a landing aircraft is the process of taxiing from the runway to the stand (parking position) and for a departing aircraft the process of taxiing from the stand to the departure runway. The Ground Controller is responsible for this process. His task is to guide the taxiing aircraft quickly and safely from the runway to the stand or vice versa given certain time constraints. The task involves assigning a taxi route to each aircraft. However, on most airports, like Amsterdam Airport Schiphol, there are standard taxi routes defined.

At an airport, aircraft are parked at a stand on the apron, where it will unload and load passengers (via a gate or a bus) and/or freight. Taxiways connect the apron with the runways. Each runway has several runway entry and exit points, these are the points that connect the taxiways with the runway. Sometimes departing aircraft will have to wait before entering the runway and taking off. This can be done at (runway) holding points.

The taxi process depends on several other processes leading to time and location requirements. Below a short description of the related processes and requirements are given. An arriving aircraft requires permission to land from the Runway Controller. The landing time is estimated as the Estimated Time of Arrival (ETA). The Actual Time of Arrival (ATA) is the actual time the aircraft touches down on the runway. When the aircraft exits the runway, it is handed over to a Ground Controller and the aircraft immediately starts taxiing to its stand. A departing aircraft may start its engine after receiving a start-up clearance and it may leave the stand after a push-back clearance. The Start-Up Controller gives these clearances. Before doing this, information about the flight, such as the take-off runway, preferred take-off sequence and Target Time for Departure (TTD), should be available. The latter is the time the aircraft should take-off from the runway. The time the aircraft is push-backed from the stand is called the Off-Block Time (OBT). In current operations the Off-Block Time is determined based on rough estimations of the average taxi times. After the push-back, control is handed over to the Ground Controller, who is responsible during taxiing. When the aircraft reaches the runway (holding point) the control is passed to the Runway Controller.

The Ground Controller should make sure that conflicts are avoided. A conflict occurs when aircraft are too close together and there is a possibility of a collision between (two) aircraft. There are three types of conflicts (see Figure 1):

- two aircraft cross each other (using the same taxiway intersection at the same time);
- two aircraft trailing each other. This will cause a conflict if the aircraft that is behind has a higher speed;

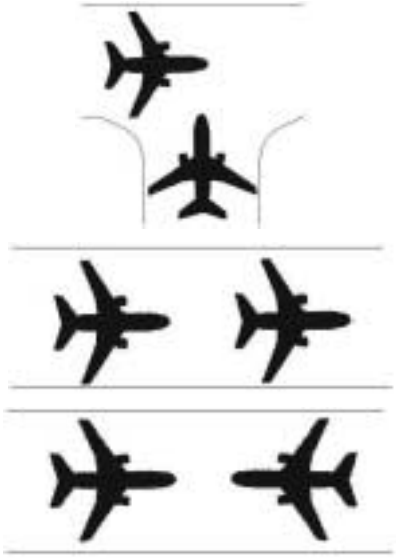

Fig. 1 Illustration of the three different types of conflicts: crossing (top), trailing (middle) and towards each other (bottom).

- two aircraft taxiing towards each other on the same piece of taxiway.

3 Model

In this section a model of the taxi planning problem is given. Given a route and an arrival or departure time for each aircraft, it should be decided at which time an aircraft should reach a particular point at the airport such that no conflicts occur and all aircraft meet the time requirement. To improve the efficiency, the waiting times while taxiing should be minimised.

The taxiways on an airport can be represented by a directed graph $G = (V, A)$ with V the set of nodes and A the set of arcs. Each node represents an intersection of taxiways. An arc represents a taxiway between two intersections. Additionally, a function $l : A \rightarrow \mathbb{R}^+$ denotes the length of an arc. Figure 2 illustrates the graph used for Amsterdam Airport Schiphol.

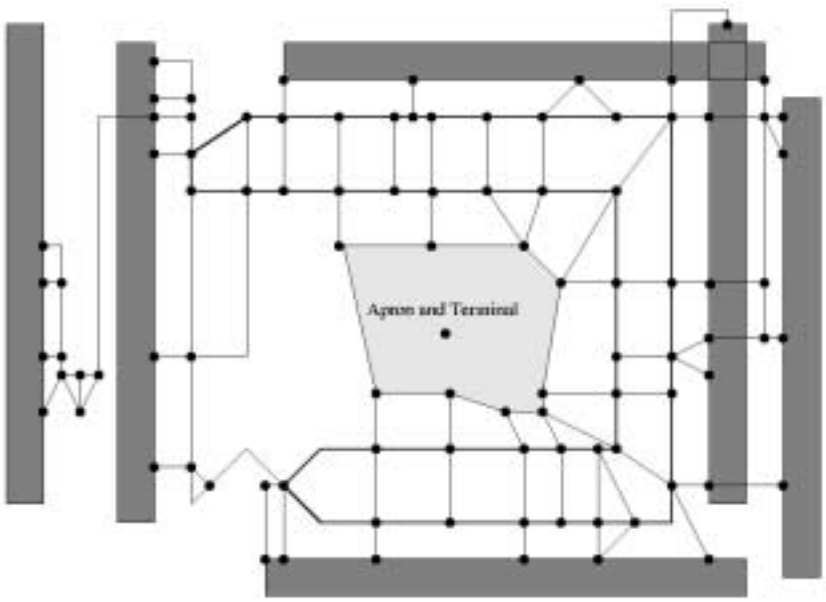

Fig. 2 A schematic graph representation of the taxiways at Amsterdam Airport Schiphol. The dark grey areas denote the runways and the light gray area the apron.

Let $\mathcal{A} = \{1, \ldots, n\}$ be the set of aircraft that are going to be planned. This set can be partitioned into the set of arriving aircraft $\mathcal{A}^{\text{arr}} \subset \mathcal{A}$ and the set of departing aircraft $\mathcal{A}^{\text{dep}} \subset \mathcal{A}$. Let $R = \{R_1, R_2, \ldots, R_n\}$ be the set of taxi routes, where R_i is the taxi route of aircraft i. A taxi route R_i is an ordered set of nodes $(u^i_1, u^i_2, u^i_3, \ldots, u^i_{k_i})$, with $u^i_j \in V$ for $j = 1, \ldots, k_i$ such that $(u^i_j, u^i_{j+1}) \in A$ for $j = 1, \ldots, k_i - 1$. For the sake of simplificity, the notation $(u, v) \in R_i$ is used if the arc (u, v) is part of the route of aircraft i, and also $u \in R_i$ is used to denote that node u is in the route of aircraft i.

For each node on a route, it has to be decided at which time the aircraft should reach that node.

This can be done by assigning a time $t_{iu} \in \mathbb{R}^+$ for each $i \in \mathcal{A}$ and $u \in R_i$. This variable denotes the scheduled time at which aircraft i reaches node u belonging to its route.

Since the same node can be contained in two (or more) aircraft's routes, it needs to be decided in which order the aircraft pass such a node. Therefore, two types of 0-1 decision variable are introduced. Variable $y_{iju} = 1$ if aircraft i visits node u directly before aircraft j and 0 otherwise. Variable $z_{iju} = 1$ if aircraft i reaches node u before aircraft j and 0 otherwise. Since one aircraft is first and another is last for a certain node, a dummy aircraft can be introduced to proceed and succeed all other aircraft. This dummy aircraft is denoted as aircraft 0 and $\mathcal{A}^0 = \mathcal{A} \cup \{0\}$.

Next, the following constraints are introduced.

$$\sum_{i \in \mathcal{A}^0} y_{iju} = 1 \qquad \forall j \in \mathcal{A}^0,\ \forall u\ \in R_i \cap R_j \tag{1}$$

$$\sum_{j \in \mathcal{A}^0} y_{iju} = 1 \qquad \forall i \in \mathcal{A}^0,\ \forall u\ \in R_i \cap R_j \tag{2}$$

$$t_{ju} > y_{iju} t_{iu} \qquad \forall i, j \in \mathcal{A},\ i \neq j, \forall u \in R_i \cap R_j \tag{3}$$

$$t_{iu} > y_{0ju} t_{ju} \qquad \forall i, j \in \mathcal{A},\ i \neq j, \forall u \in R_i \cap R_j \tag{4}$$

$$t_{iu} > y_{i0u} t_{ju} \qquad \forall i, j \in \mathcal{A},\ i \neq j, \forall u \in R_i \cap R_j \tag{5}$$

$$z_{iju} \geq y_{iju} \qquad \forall i, j \in \mathcal{A}^0, i \neq j, \forall u \in R_i \cap R_j \tag{6}$$

$$z_{jiu} \geq y_{0ju} \qquad \forall i, j \in \mathcal{A}, i \neq j, \forall u \in R_i \cap R_j \tag{7}$$

$$y_{0ju} + z_{iju} \leq 1 \qquad \forall i, j \in \mathcal{A}, i \neq j, \forall u \in R_i \cap R_j \tag{8}$$

$$z_{jiu} \geq y_{i0u} \qquad \forall i, j \in \mathcal{A}, i \neq j, \forall u \in R_i \cap R_j \tag{9}$$

$$y_{i0u} + z_{iju} \leq 1 \qquad \forall i, j \in \mathcal{A}, i \neq j, \forall u \in R_i \cap R_j \tag{10}$$

$$z_{kju} \geq y_{iju} + z_{kiu} - 1 \qquad \forall i, j, k \in \mathcal{A}, i \neq j \neq k,\ \forall u \in R_i \cap R_j \cap R_k \tag{11}$$

$$y_{iju}, z_{iju} \in \{0, 1\} \qquad \forall i, j \in \mathcal{A}, i \neq j, \forall u \in R_i \cap R_j \tag{12}$$

Constraint (1) and (2) assure that each aircraft has only one immediate successor and only one immediate predecessor (which may be a dummy aircraft) at node u. Constraint (3) assures that if $y_{iju} = 1$, aircraft i actually reaches node u before aircraft j. Constraint (4) assures that if $y_{0ju} = 1$, aircraft j actually reaches node u before all other aircraft. Constraint (5) assures that if $y_{i0u} = 1$, aircraft i actually reaches node u after all other aircraft. Constraints (1) - (5) guarantee that $y_{iju} = 1$ if and only if aircraft j is the first aircraft reaching node u after aircraft i. If aircraft j follows aircraft i directly at node u ($y_{iju} = 1$), constraint (6) assures that $z_{iju} = 1$. If aircraft j is the first aircraft reaching node u ($y_{0ju} = 1$), constraint (7) assures that $z_{jiu} = 1$ for each aircraft

i reaching node u. Constraint (8) states that if aircraft j is the first aircraft to reach node u, then aircraft j will not follow any other aircraft. Similar to constraints (7) and (8), constraints (9) and (10) state that if aircraft i is the last aircraft to reach node u, aircraft i will follow any other aircraft j and no other aircraft j will follow i. Constraint (11) assures that if aircraft j follows aircraft i immediately at node u, then aircraft j also follows all aircraft k that are followed by aircraft i at node u. Constraints (6) - (11) guarantee that $z_{iju} = 1$ if and only if aircraft j reaches node u later than aircraft i.

The scheduled times should be compliant with the possible taxi speeds of an aircraft. Therefore, for each arc on a route R_i, constants $d_{iuv}^{\min}$ (and $d_{iuv}^{\max}$) are derived denoting the minimum (maximum) time needed for aircraft i to taxi along the edge (u, v) of its route. These constants can be calculated using the minimum (maximum) speed of aircraft i, and the length of the arc, where the minimum and maximum speed can vary if the arc is a turn or a straight segment. This yields the following inequalities:

$$t_{iu} + d_{iuv}^{\min} \leq t_{iv} \leq t_{iu} + d_{iuv}^{\max}, \quad \forall (u, v) \in R_i, i = 1, \ldots, n \tag{13}$$

Although a maximum taxi time for an arc is not strictly necessary, the introduction will be explained later on in Section 4.

To avoid conflicts, a separation distance $d_{\text{sep}} = 200m$ is defined to prevent the aircraft's wing tip from touching the nose or rear of another aircraft on a taxiway intersection. In the case of two trailing aircraft, it also protects the second aircraft from the jet blast coming from the first aircraft's engine.

It will be sufficient to ensure separation at the nodes (intersections) only: If two aircraft travel the same edge (taxiway) between two nodes, the separation at these nodes will be enforced. Assuming that aircraft taxi at the same speed between two nodes, the separation will also be achieved on the edge. Therefore, the separation time is used $s_{iju} = (t_{iv} - t_{iu})/l(u, v) \cdot d_{\text{sep}}$ denoting the minimal time difference between aircraft i and j reaching node u, if aircraft i reaches node u before aircraft j. This results in the following constraint for the separation:

$$z_{iju} t_{ju} \geq z_{iju}(t_{iu} + s_{iju}), \qquad \forall i, j \in \mathcal{A}, i \neq j, \forall u \in R_i \cap R_j \, . \tag{14}$$

Since the scheduled times t are only computed at the nodes of a route, an additional constraint needs to be introduced to prevent aircraft from overtaking on a segment.

$$z_{iju} - z_{ijv} = 0, \qquad \forall i, j \in \mathcal{A}, \forall (u, v) \in R_i \cap R_j \, . \tag{15}$$

This constraint implies that if aircraft i reaches node u earlier than aircraft j, then it should reach node v earlier than aircraft j, for both aircraft using taxiway (u, v).

To ensure that two aircraft do not cross each other on the same segment, the following constraint is used:

$$z_{iju} - z_{ijv} = 0, \qquad \forall i, j \in \mathcal{A}, \forall (u, v) \in R_i \text{ with } (v, u) \in R_j \, . \tag{16}$$

If aircraft i reaches node u earlier than aircraft j, it should reach node v earlier than aircraft j, when aircraft i is taxiing from u to v and aircraft j from v to u.

There is a maximum number of aircraft that can wait at a holding point. Holding points are modelled as series of successive (nodes and) arcs A^{hold} with length 0. The number of arcs is equal to the capacity of the holding point. There is a maximum of one aircraft at a time at a single edge of a holding point. So,

$$z_{iju} t_{ju} > z_{iju} t_{iv}, \qquad \forall i, j \in A, \forall (u, v) \in R_i \cap R_j \cap A^{\text{hold}} \, . \tag{17}$$

If aircraft i reaches node u before aircraft j, and edge (u, v) is part of a holding point, then aircraft i should reach node v, before aircraft j reaches node u.

An arriving aircraft must immediately (clear the runway and) start taxiing when it arrives, i.e. at the estimated time of arrival (ETA_i), so:

$$t_{iu_1^i} = ETA_i, \qquad \forall i \in \mathcal{A}^{\text{arr}} \, . \tag{18}$$

A departing aircraft cannot start taxiing before its Off Block Time (OBT_i):

$$t_{iu_1^i} \geq OBT_i, \qquad \forall i \in \mathcal{A}^{\text{dep}} \, . \tag{19}$$

A departing aircraft must have finished taxiing (and thus must have reached the runway) before its target time of departure (TTD_i):

$$t_{iu_{k_i}^i} \leq TTD_i, \qquad \forall i \in \mathcal{A}^{\text{dep}} \, . \tag{20}$$

Recall that u_1^i is the first node of the route of aircraft i and $u_{k_i}^i$ the last node of the route of aircraft ii. It can be preferred to adapt to the departure schedule in a stricter way. In constraint (20) the inequality can then be replaced by an equality. Optionally, it can be required that the departing aircraft take-off exactly in the order determined by the departure planning. This order can be derived from the TTD's. To make this possible, all the departing aircraft who share the last node (runway exit point) of their taxi route should reach the node in the right order. Constraints can be

added to ensure this. So if, according to the departure planning, aircraft j is the first aircraft to enter the runway at node u after aircraft i, constraints stating that $y_{iju} = z_{iju} = 1$ and $y_{jiu} = z_{jiu} = 0$ are added.

Clearly, constraints (3), (4), (5), (14), and (17) are non-linear. However, these can be linearised by introducing a large number M (see e.g. [10, 12]):

$$t_{iu} < t_{ju} + (1 - y_{iju})M \qquad \forall i, j \in \mathcal{A}, i \neq j, \forall u \in R_i \cap R_j \tag{21}$$

$$t_{iu} > t_{ju} + (1 - y_{0ju})M \qquad \forall i, j \in \mathcal{A}, i \neq j, \forall u \in R_i \cap R_j \tag{22}$$

$$t_{iu} > t_{ju} + (1 - y_{i0u})M \qquad \forall i, j \in \mathcal{A}, i \neq j, \forall u \in R_i \cap R_j \tag{23}$$

$$t_{ju} - t_{iu} \geq s_{iju} - (1 - z_{iju})M \qquad \forall i, j \in \mathcal{A}, i \neq j, \forall u \in R_i \cap R_j \tag{24}$$

$$t_{ju} > t_{iv} - (1 - z_{iju})M \qquad \forall i, j \in \mathcal{A},\ \forall (u, v) \in R_i \cap R_j \cap A^{\text{hold}} \tag{25}$$

Since y and z are binary variables, the above constraints are only effective if these variables are 1. Otherwise, the M, if chosen large enough, causes these constraints to be redundant (e.g. stating that $t_{iu} - t_{ju} < M$). To guarantee that M is large enough, one could for example take the sum of the latest possible times that each aircraft is finished taxiing.

In order to minimise the use of the airport resources, the goal is to minimise the sum of the total taxi times for all aircraft:

$$\min_t \sum_{i \in \mathcal{A}} (t_{iu^i_{k_i}} - t_{iu^i_1}) \,, \tag{26}$$

where t is the vector denoting all t_{iu} with $i \in \mathcal{A}$ and $u \in R_i$. This objective function will ensure that the taxiways are used as little time as possible. Using this objective function might cause undesirable schedules with respect to the departure planning: a departing aircraft can arrive long before its TTD at the runway (holding point), therefore causing congestion. This can be avoided by also incorporating the time that departing aircraft are waiting at the runway (entry point), thus minimising the total time spend from starting taxiing until the TTD. For arriving aircraft no such a problem occurs. This gives the following objective function:

$$\min_t \sum_{i \in \mathcal{A}^{\text{dep}}} (TTD_i - t_{iu^i_1}) + \sum_{i \in \mathcal{A}^{\text{arr}}} (t_{iu^i_{k_i}} - t_{iu^i_1}) \tag{27}$$

Note that the constants TTD_i can be removed.

4 Optimisation

The model of the taxi problem described in the previous section, is formulated as a Mixed Integer Programming (MIP) problem, and it can be shown that the decision variant of the taxi problem is NP-complete by reduction from a job-shop scheduling problem with release dates which is known to be NP-complete [4, 5]. The nodes of the graph representing the taxiways correspond with the machines of the job-shop and the required separations for the nodes are the processing times.

To reduce the computation time, some additional pre-processing can be performed. Analysis of the optimal taxi schedules of different problem instances, show that very low taxi speeds rarely occur. In these instances speeds lower than 8 knots ($\approx 15\mathrm{km/h}$) were not observed. Most of the time aircraft were taxiing at their maximum speed or at the maximum speed of a slower aircraft in front of them. Aircraft standing still on taxiways did not occur. The above suggests that a minimal speed can be introduced. If chosen low enough, this will not affect the optimal schedule. Without a minimum speed, $d_{iuv}^{\max}$ will be infinite. A minimum speed will therefore impose stricter bounds on variables and could thus reduce computation time. To reduce the computation time without introducing infeasibility and suboptimal solutions, a minimum speed of 5 knots ($\approx 9\mathrm{km/h}$) seems to be appropriate. For the optimal solution it needs to be verified that the introduced bounds are not tight.

Using maximum and minimum taxi speeds, the earliest and latest possible time an aircraft is able to reach a node of its route can be computed. If the latest time an aircraft i is able to reach node u, is earlier than the earliest time aircraft j is allowed to reach the same node, then it is known for sure that in every feasible taxi schedule aircraft i will reach node u before aircraft j. This is equivalent to fixing $z_{iju} = 1$ and $z_{jiu} = 0$.

Using CPLEX 7.5 [8] it is possible to solve within reasonable time instances of the taxi problem with the Schiphol lay-out up to approximately 20 aircraft taxiing more or less simultaneously.

A busy day at Schiphol consists of approximately 1000 aircraft. A taxi schedule for such a problem size can be constructed by combining several smaller instances using so-called rolling horizon algorithms. For this problem three different variants of rolling horizon algorithms are implemented.

Rolling Horizon Variant 1

For the first variant, the planning period is split in a series of disjunct time intervals of equal length T. Aircraft are assigned to an interval, based on the earliest possible time they can start taxiing (OBT for departing aircraft and ETA for arriving aircraft). The aircraft are scheduled per interval,

and each interval is treated in chronological order. In each iteration the aircraft in an interval are scheduled using the MIP formulation and CPLEX, taking aircraft planned in earlier intervals into account. The schedules of the aircraft that are planned in an earlier iteration are taken as fixed. This means the values of the time decision variables, representing the times these aircraft reaches nodes, will be fixed.

Rolling Horizon Variant 2
The second variant is similar to the previous one, but by fixing fewer variables better schedules are expected. Consider an aircraft that was grouped in a previous interval but that will reach (some) nodes of its route in the current interval. The part of the route of this aircraft that lies within this time interval will be replanned for the current interval. The times of the nodes that were scheduled to be reached before the start of the current interval are fixed.

Rolling Horizon Variant 3 (sliding window)
The third variant, is often referred to as a sliding window algorithm. In the previous variants (partly) fixed aircraft and aircraft dependent of these fixed aircraft, are not scheduled optimally. This can cause undesirable situations for these aircraft. A sliding window algorithm tries to spread the undesirable effects among all aircraft. This is done by sorting the list of aircraft according to the earliest time they are able to start taxiing. In every iteration m aircraft are considered. m will be called the size of the sliding window. In the first iteration aircraft $1, \ldots, m$ are scheduled. After that, the first one of the m aircraft is fixed. In the next iteration, the sliding window is advanced one aircraft, and so aircraft $2, \ldots, m+1$ are scheduled. This is repeated until finally aircraft $n-m+1, \ldots, n$ are planned.

The overall schedule is not necessarily optimal. When the interval length or the size of the sliding window is chosen well, it will be of reasonably good quality: This can be reasoned as follows. First, the constraints are satisfied for all aircraft, so the schedule is valid, safe, and complies with the departure and arrival schedules. Second, aircraft separated by a long period of time are nearly independent: an aircraft taxi's schedule will usually not be influenced by an aircraft taxiing two hours earlier. Aircraft taxiing within a short time interval (and sharing parts of their routes) are dependent. Optimality cannot be guaranteed for the fixed aircraft and the aircraft that are dependent on these aircraft. A too small interval length would imply a relative large share of fixed aircraft. It might be impossible to schedule other aircraft, complying with the fixed aircraft and thus cause infeasibility. If chosen too large, the size of the MIP models will be large, which may result in large computation times per iteration.

To compare different schedules, the ideal time for each aircraft is defined. This is the taxi time

required when the aircraft is the only aircraft on the airport, and hence no delays are introduced as a result of other aircraft. This ideal taxi time is a lower bound on the taxi time of an aircraft in a feasible schedule.

5 Results

To analyse the effects of pre-processing and to determine the length of the time intervals T and the size of the sliding window m, 12 instances of approximately 30 minutes at Amsterdam Airport Schiphol were constructed. The results are presented in Table 1. All computations are performed on a Compaq computer with an Intel Pentium III processor (866Mhz) and a Linux operating system. The results show that instances up to approximately 20 aircraft can be scheduled within reasonable times. As soon as the number of aircraft becomes too large (Instances 4, 8, and 12), the optimal solution cannot be found within 1000 seconds. Except for Instances 1 and 2, pre-processing yields significant improvement in the computation times.

The problem instance used to test the complete algorithm is a typical day at Amsterdam Airport Schiphol. It consists of 406 aircraft: 189 arriving and 217 departing aircraft on a Sunday in September 2002 between 11.30 and 17.00 hours. This specific day can be considered as a busy one and is therefore a good test case for the model.

Objective function (27), a minimum taxi speed of 5 knots, and a constant separation distance of 200 meters were used. Pre-processing by fixing the known sequence variables is applied. Maintaining departure sequence, fixing TTD's and holding points were not used.

The individual ideal taxi times can be used as a lower bound for the obtained solutions. For this problem instance, the average individual ideal taxi time is 161 seconds with a standard deviation of 87 seconds. As a reference for the obtained results, a simulation version of this model is used to model the current situation. In this simulation model, the arriving aircraft start taxiing on the ETA's and the departing aircraft on the OBT's. Potential conflicts are solved on a first-come-first-serve basis. According to the simulation, in the current situation the average aircraft is 20% percent slower than its individual ideal schedule (total taxi time / ideal taxi time = 1.20).

The first variant of the rolling horizon algorithm yields no feasible solution for the interval length T equal to 10 minutes. This indicates that this interval length is too small. This yields a relative large share of fixed aircraft. This leads to a situation in which it is impossible to schedule other aircraft, compliant with the fixed aircraft. An interval length T of 15 minutes gives feasible solutions. The algorithm performs 24 iterations and on average 17 aircraft are scheduled each iteration. The total computation time was 1300 seconds. One iteration took longer than 500 seconds. The average computing time of the remaining iterations was around 30 seconds. In Table 2 the performance is given. The average total taxi time in the schedule is only 2 seconds higher than the individual ideal taxi time. When waiting at the runway (until the TTD) is included the difference is 4 seconds. On

Inst.	# arrivals	# departures	# int. var	time (s) no pre-proc.	time (s) with pre-proc.
1	11	14	3480	18.35	39.35
2	14	17	4216	33.5	33.5
3	15	24	4148	27.18	16.75
4	19	31	11104	(***)	(***)
5	20	12	5268	20.88	11.64
6	19	14	4932	74.39	36.6
7	18	16	5324	37.14	6.11
8	24	30	13252	(***)	(***)
9	22	9	5436	590	96.76
10	17	13	4756	18.45	9.7
11	19	4	2872	6.08	1.95
12	21	3	3964	(***)	(***)

Table 1 The results of 12 instances. The second and third column denote the number of arriving and departing aircraft, respectively. The fourth column denotes the number of integer variables in the MIP formulation. The last two columns denote the computation times in seconds obtained with and without pre-processing, respectively. The instances denote approximately 30 minutes at Amsterdam Airport Schiphol. The (***) indicate that the problem instance was not solved within 1000 seconds.

average, departing aircraft taxi almost a minute longer than arriving aircraft, but this is simply caused by the longer taxi routes of departing aircraft. The average aircraft is only 2% slower than its individual ideal schedule (in the current situation, represented by the simulation data, this number is 20%) .

	Variant 1		Variant 2		Variant 3	
	Mean	Std	Mean	Std	Mean	Std
Total taxi time (sec)	163	87	163	87	163	88
Total taxi time until TTD (sec)	165	90	165	90	165	89
Individual ideal taxi Time (sec)	161	87	161	87	161	87
Total taxi time / ideal taxi time	1.02	0.11	1.02	0.10	1.02	0.08

Table 2 Performance using the three different variants displaying the mean taxi time and the standard deviation.

In Figure 3 the delay until the TTD is depicted. 339 of the 406 aircraft were planned according to their individual ideal schedule. The remaining 67 spend on average almost 10 seconds more on taxiing (or waiting at the runway) than in their individual ideal schedules. The largest delay was 79 seconds.

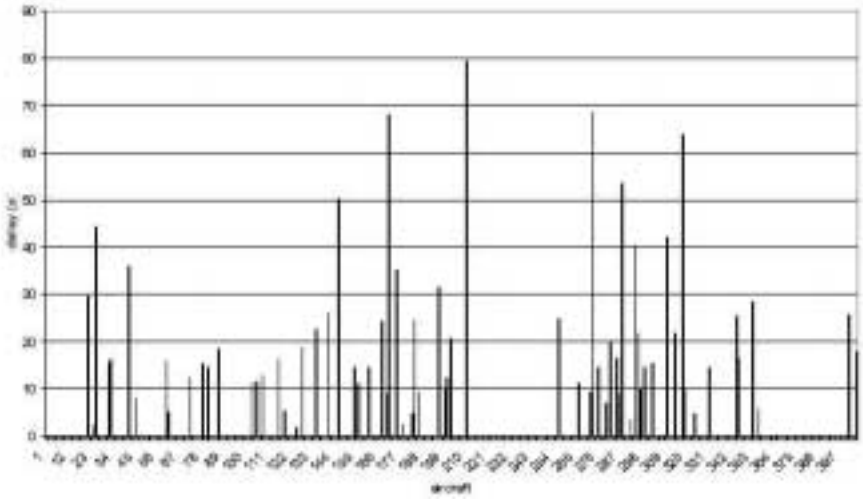

Fig. 3 Graph of the delay per aircraft in seconds in the obtained solution with the first rolling horizon variant. The aircraft are numbered in chronological order of starting times.

For the second rolling horizon variant, an interval length T of 10 minutes also causes infeasibility. With T is 15 minutes, the algorithm performed 24 iterations. The total computation time was 844 seconds, which is on average over 30 seconds per iteration.

The performance statistics for Variant 2 as listed in Table 2 do hardly differ from those for Variant 1. In Figure 4 the delay until the TTD is depicted. 338 of the 406 aircraft were planned according to their individual ideal schedule. This is one aircraft less than with Variant 1. The other 68 spent on average almost 9 seconds more on taxiing (or waiting at the runway) than in their individual ideal schedules. Although one aircraft more is delayed, the average delay is lower. The largest delay was 79 seconds, just as with Variant 1.

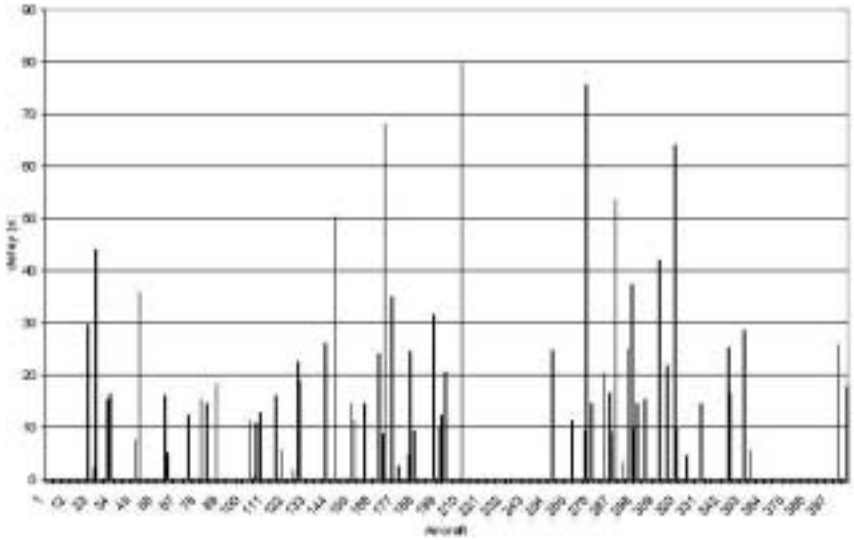

Fig. 4 Graph of the delay per aircraft in seconds in the obtained solution with the second rolling horizon variant. The aircraft are numbered in chronological order of starting times.

With the sliding window algorithm (Variant 3) the number of iterations and therefore the computation time will be much larger than with the other methods. When using a sliding window size (m) of 15 aircraft, the computation time is 9453 seconds. The average per iteration is 23 seconds. As expected the delay is more equally spread over the aircraft than with the other methods. This can be seen by the lower standard deviation of the ratio between total and ideal taxi.

Figure 5 shows that the delay is more equally spread. Now 332 of the 406 aircraft are taxiing according to their individual ideal schedule (compared to 339 and 338 using the other methods). The average delay of the other aircraft now only is 8 seconds (9 and 10 seconds with the other methods). The largest delay occurring now was 69 seconds, 10 seconds shorter than with the other methods.

Fifteen minutes seem to be a good choice as time interval for the first two methods. A period of ten minutes causes infeasibility, where longer periods will generally result in larger computation

times. Fifteen aircraft seems to be a good choice as sliding window size for the same reason. All methods have the disadvantage that the computation time per iteration varies a lot and might be undesirable large in some cases. The schedules generated using these methods are almost of the same quality. On average, the taxi time of an aircraft is two percent longer than in its individual ideal taxi schedule, with all three methods. This is a considerable improvement from the current situation of which simulation showed that this is currently around 20%. More than 80% of the aircraft are taxiing according to their individual ideal schedule in the generated schedules. The other aircraft spend more time taxiing, but not more than 80 seconds per aircraft. Although the sliding window algorithm results in a schedule where delay is slightly more equally spread among all aircraft, the computation time of this algorithm is much larger than with the other two methods, because of the large number of iterations.

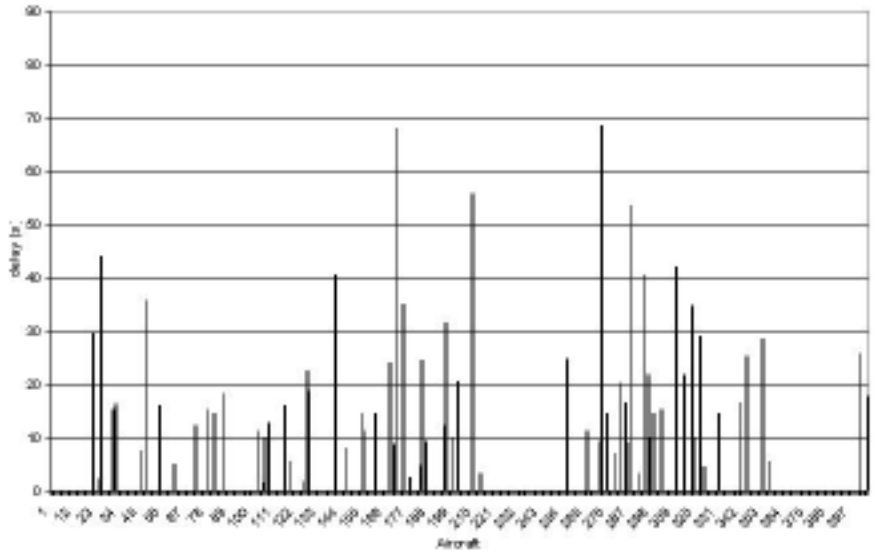

Fig. 5 Graph of the delay per aircraft in seconds in the obtained solution with the third rolling horizon variant (the sliding window). The aircraft are numbered in chronological order of starting times.

6 Conclusions

In this study three variants of an algorithm for solving the taxi planning problem are developed and investigated. These algorithms can be used for tactical taxi planning. The three variants are all rolling horizon algorithms where the problem is split-up into sets consisting of a small number of aircraft. To create a planning for a whole day, the solutions to the sub-problems are combined. The sub-problems are solved using an MIP formulation. With this formulation and pre-processing, CPLEX was able to solve instances up to 20 aircraft within reasonable time. The first two variants use fixed time intervals to split the set of aircraft to create sub-problems. The third variant is a sliding window algorithm. An advantage of rolling horizon algorithms is that without much effort the schedule can be replanned from a certain point in time to comply with an updated situation at the airport.

As a test case a busy day at Amsterdam Airport Schiphol was used. This problem instance consists of scheduling 406 aircraft in a period of $5\frac{1}{2}$ hours. Using the optimisation algorithm it was possible to reduce the average delay from 20% to 2%, where the individual ideal taxi time is used as a lower bound for the obtained solution. This reduction of taxi delay makes the taxi process more efficient, and it also reduces the fuel emission levels due to taxiing. These emissions have become a major environmental problem at busy airports.

All three variants show similar results with respect to the objective function. The first two variants, with the fixed time intervals, obtain better results with respect to the computation times than the third variant (sliding window): 844, 1300, and 9453 seconds respectively. Since this study was used to develop an initial model and algorithm, it is expected that the computation times can be reduced further. The rolling horizon approach seems suitable for a dynamic environment like the operations at an airport, where looking far ahead in time has no point since the planning can often change completely (e.g. due to delays). Moreover, this exercise demonstrates that splitting the problem into smaller sub-problems has almost no effect on the quality of the obtained overall solution.

7 References

1. A. Bolat, *Models and a genetic algorithm for a static aircraft-gate assignment problem*, Journal of the Operations Research Society **52** (2001), pp. 1107-1120.
2. Y. Cheng, *Solving push-out conflicts in apron taxiways of airports by a network based simulation*, Computer Industrial Engineering **2** (1998), pp. 351-369.
3. European CDM website, *Collaborative Decision Making for the ATM Industry*, `http://www.euro-cdm.org/`
4. M.R. Garey, and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*,W.H. Freeman, New York, 1979.
5. M.R. Garey, D.S. Johnson, and R. Sethi, *The complexity of flowshop and jobshop scheduling*, Mathematics of Operations Research, **1** (1976), pp. 117-129.
6. H.H. Hesselink and N. Basjes, *Mantea Departure Sequencer: Increasing Airport Capacity by Planning Optimal Sequences*, 2nd USA/Europe Air Traffic Management R&D Seminar, 1998.
7. H. Idris, J-P. Clarke, R. Bhuva, L. Kang, *Queuing model for taxi time estimation*, ATC Quarterly, **10**(2002), pp. 1-22.
8. ILOG, CPLEX 7.5.0, 2001
9. F. Neuman and H. Erzberger, *Analysis of sequencing and scheduling methods for arrival traffic*, NASA Technical Memorandum 102795, Ames Research Center, Moffet Field, 1990.
10. C.H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Englewood Cliffs, 1982.
11. D.E. Pitfield, A.S. Brooke and E.A. Jerrard, *A Monte-Carlo simulation of potentially conflicting ground movements at a new international airport*, Journal of Air Transport Management **4**(1998), pp. 3-9.
12. H.P. Williams, *Model Building in Mathematical Programming*, 3rd edition, John Wiley and Sons, Chichester, 1993.